Jul 31 2023

NETMANAGEIT

Intelligence Report

MAR-10454006-r2.v1

SEASPY Backdoor

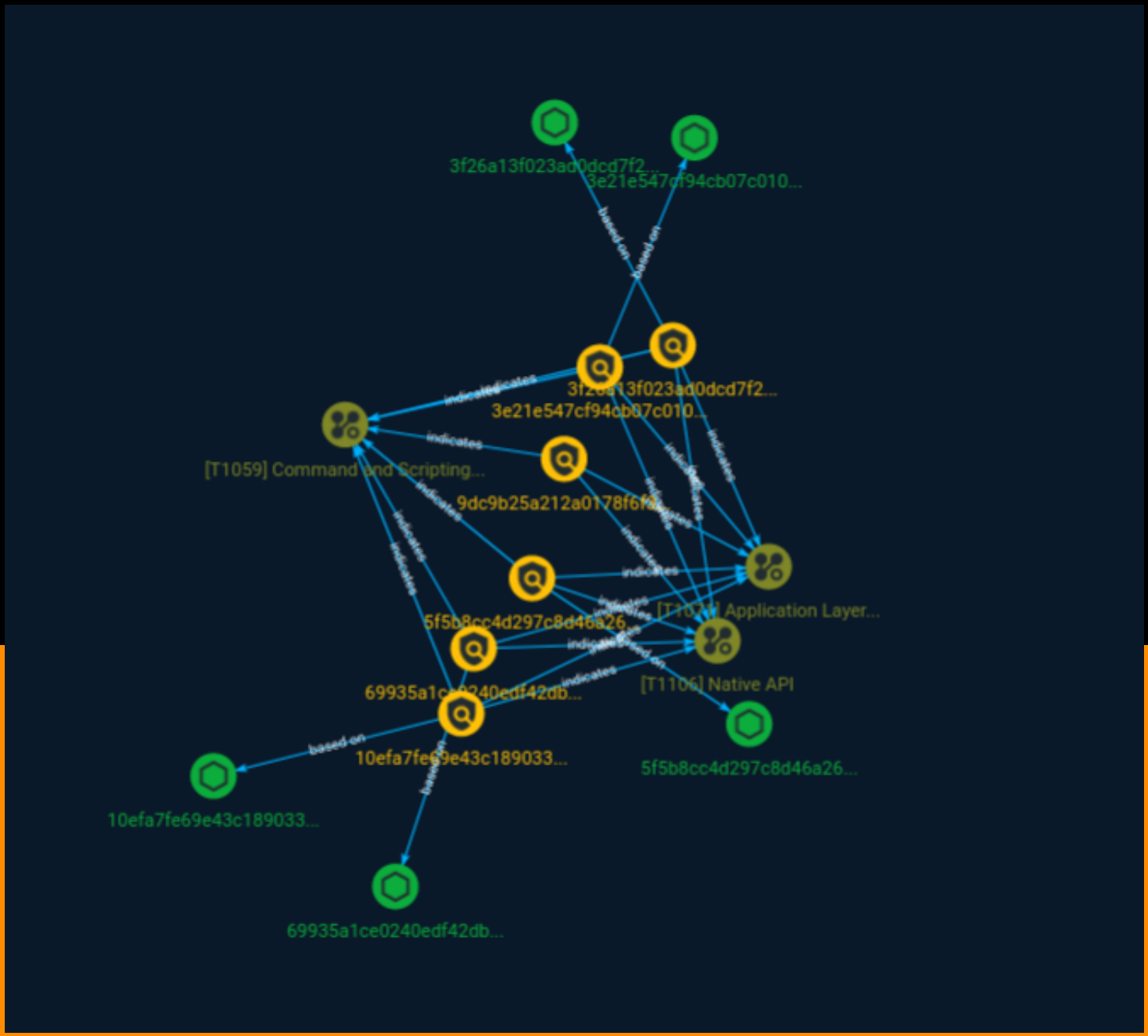

Table of contents

Overview

Entities

Observables

External References

Overview

Description

CISA obtained two SEASPY malware samples. The malware was used by threat actors exploiting CVE-2023-2868, a former zero-day vulnerability affecting versions 5.1.3.001-9.2.0.006 of Barracuda Email Security Gateway (ESG). SEASPY is a persistent and passive backdoor that masquerades as a legitimate Barracuda service “BarracudaMailService” that allows the threat actors to execute arbitrary commands on the ESG appliance.

Confidence

This value represents the confidence in the correctness of the data contained within this report.

15 / 100

Indicator

Name
69935a1ce0240edf42dbe24535577140601bcf3226fa01e4481682f6de22d192
Pattern Type
stix
Pattern
[file:hashes.'SHA-256' = '69935a1ce0240edf42dbe24535577140601bcf3226fa01e4481682f6de22d192']
Name
10efa7fe69e43c189033006010611e84394569571c4f08ea1735073d6433be81
Pattern Type
stix
Pattern
[file:hashes.'SHA-256' = '10efa7fe69e43c189033006010611e84394569571c4f08ea1735073d6433be81']
Name

9dc9b25a212a0178f6f3d7789f8be10f57bca164

Pattern Type

yara

Pattern

rule CISA_10452108_01 : SEASPY backdoor communicates_with_c2 installs_other_components { meta: Author = "CISA Code & Media Analysis" Incident = "10452108" Date = "2023-06-20" Last_Modified = "20230628_1000" Actor = "n/a" Family = "SEASPY" Capabilities = "communicates-with-c2 installs-other-components" Malware_Type = "backdoor" Tool_Type = "unknown" Description = "Detects malicious Linux SEASPY samples" SHA256_1 = "3f26a13f023ad0dcd7f2aa4e7771bba74910ee227b4b36ff72edc5f07336f115" SHA256_2 = "69935a1ce0240edf42dbe24535577140601bcf3226fa01e4481682f6de22d192" SHA256_3 = "5f5b8cc4d297c8d46a26732ae47c6ac80338b7be97a078a8e1b6eefd1120a5e5" SHA256_4 = "10efa7fe69e43c189033006010611e84394569571c4f08ea1735073d6433be81" strings: $s0 = { 2e 2f 42 61 72 72 61 63 75 64 61 4d 61 69 6c 53 65 72 76 69 63 65 20 65 74 68 30 } $s1 = { 75 73 61 67 65 3a 20 2e 2f 42 61 72 72 61 63 75 64 61 4d 61 69 6C 53 65 72 76 69 63 65 20 3c 4e 65 74 77 6f 72 6b 2d 49 6e 74 65 72 66 61 63 65 } $s2 = { 65 6e 74 65 72 20 6f 70 65 6e 20 74 74 79 20 73 68 65 6c 6c } $s3 = { 25 64 00 4e 4f 20 70 6f 72 74 20 63 6f 64 65 } $s4 = { 70 63 61 70 5f 6c 6f 6f 6b 75 70 6e 65 74 3a 20 25 73 } $s5 = { 43 68 69 6c 64 20 70 72 6f 63 65 73 73 20 69 64 3a 25 64 } $s6 = { 5b 2a 5d 53 75 63 63 65 73 73 21 } $a7 = { bf 90 47 90 ec 18 fe e3 83 e2 a9 f7 8d 85 18 1d } $a8 = { 81 35 1e f0 94 ab 2a ba 5d f0 37 76 69 19 9f 1e } $a9 = { 6a 8e c7 89 ce c1 fe 64 78 a6 e1 c5 fe 03 d1 a7 } $a10 = { c2 ff d1 0d 24 23 ec c0 57 f9 8d 4b 05 34 41 b8 } condition: uint32(0) == 0x464c457f and (all of ($s*)) or (all of ($a*)) }

Name

5f5b8cc4d297c8d46a26732ae47c6ac80338b7be97a078a8e1b6eefd1120a5e5

Pattern Type

stix

Pattern

[file:hashes.'SHA-256' = '5f5b8cc4d297c8d46a26732ae47c6ac80338b7be97a078a8e1b6eefd1120a5e5']

Name

3e21e547cf94cb07c010fe82d6965e5bd52dbdd9255b4dd164f64addfaa87abb

Pattern Type

stix

Pattern

[file:hashes.'SHA-256' = '3e21e547cf94cb07c010fe82d6965e5bd52dbdd9255b4dd164f64addfaa87abb']

Name

3f26a13f023ad0dcd7f2aa4e7771bba74910ee227b4b36ff72edc5f07336f115

Description

stack_string

Pattern Type

stix

Pattern

[file:hashes.'SHA-256' = '3f26a13f023ad0dcd7f2aa4e7771bba74910ee227b4b36ff72edc5f07336f115']

Attack-Pattern

Name
Native API
ID
T1106
Description
Adversaries may interact with the native OS application programming interface (API) to execute behaviors. Native APIs provide a controlled means of calling low-level OS services within the kernel, such as those involving hardware/devices, memory, and processes. (Citation: NT API Windows)(Citation: Linux Kernel API) These native APIs are leveraged by the OS during system boot (when other system components are not yet initialized) as well as carrying out tasks and requests during routine operations. Native API functions (such as `NtCreateProcess`) may be directed invoked via system calls / syscalls, but these features are also often exposed to user-mode applications via interfaces and libraries.(Citation: OutFlank System Calls)(Citation: CyberBit System Calls)(Citation: MDSec System Calls) For example, functions such as the Windows API `CreateProcess()` or GNU `fork()` will allow programs and scripts to start other processes.(Citation: Microsoft CreateProcess)(Citation: GNU Fork) This may allow API callers to execute a binary, run a CLI command, load modules, etc. as thousands of similar API functions exist for various system operations. (Citation: Microsoft Win32)(Citation: LIBC)(Citation: GLIBC) Higher level software frameworks, such as Microsoft .NET and macOS Cocoa, are also available to interact with native APIs. These frameworks typically provide language wrappers/abstractions to API functionalities and are designed for ease-of-use/portability of code.(Citation: Microsoft NET)(Citation: Apple Core Services)(Citation: MACOS Cocoa)(Citation: macOS Foundation) Adversaries may abuse these OS API functions as a means of executing behaviors. Similar to [Command and Scripting Interpreter](https://attack.mitre.org/techniques/T1059), the native API and its hierarchy of interfaces provide mechanisms to interact with and utilize

various components of a victimized system. While invoking API functions, adversaries may also attempt to bypass defensive tools (ex: unhooking monitored functions via [Disable or Modify Tools](https://attack.mitre.org/techniques/T1562/001)).

Name

Command and Scripting Interpreter

ID

T1059

Description

Adversaries may abuse command and script interpreters to execute commands, scripts, or binaries. These interfaces and languages provide ways of interacting with computer systems and are a common feature across many different platforms. Most systems come with some built-in command-line interface and scripting capabilities, for example, macOS and Linux distributions include some flavor of [Unix Shell](https://attack.mitre.org/techniques/T1059/004) while Windows installations include the [Windows Command Shell](https://attack.mitre.org/techniques/T1059/003) and [PowerShell](https://attack.mitre.org/techniques/T1059/001). There are also cross-platform interpreters such as [Python](https://attack.mitre.org/techniques/T1059/006), as well as those commonly associated with client applications such as [JavaScript](https://attack.mitre.org/techniques/T1059/007) and [Visual Basic](https://attack.mitre.org/techniques/T1059/005). Adversaries may abuse these technologies in various ways as a means of executing arbitrary commands. Commands and scripts can be embedded in [Initial Access](https://attack.mitre.org/tactics/TA0001) payloads delivered to victims as lure documents or as secondary payloads downloaded from an existing C2. Adversaries may also execute commands through interactive terminals/shells, as well as utilize various [Remote Services](https://attack.mitre.org/techniques/T1021) in order to achieve remote Execution. (Citation: Powershell Remote Commands)(Citation: Cisco IOS Software Integrity Assurance - Command History)(Citation: Remote Shell Execution in Python)

Name

Application Layer Protocol

ID

T1071

Description

Adversaries may communicate using OSI application layer protocols to avoid detection/network filtering by blending in with existing traffic. Commands to the remote system, and often the results of those commands, will be embedded within the protocol traffic between the client and server. Adversaries may utilize many different protocols, including those used for web browsing, transferring files, electronic mail, or DNS. For connections that occur internally within an enclave (such as those between a proxy or pivot node and other nodes), commonly used protocols are SMB, SSH, or RDP.

StixFile

Value
69935a1ce0240edf42dbe24535577140601bcf3226fa01e4481682f6de22d192
3f26a13f023ad0dcd7f2aa4e7771bba74910ee227b4b36ff72edc5f07336f115
5f5b8cc4d297c8d46a26732ae47c6ac80338b7be97a078a8e1b6eefd1120a5e5
10efa7fe69e43c189033006010611e84394569571c4f08ea1735073d6433be81
3e21e547cf94cb07c010fe82d6965e5bd52dbdd9255b4dd164f64addfaa87abb

External References

- https://otx.alienvault.com/pulse/64c7f1803ca549d1ff8cb8a0
- https://www.cisa.gov/news-events/analysis-reports/ar23-209b